

**PATENT**  
9862-000019/US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application No.: NEW Group Art Unit: Unknown  
Filing Date: February 27, 2004 Examiner: Unknown  
Applicants: Choong-Bin LIM et al. Conf. No.: Unknown  
Title: A DEVICE FOR CONTROLLING ENDPOINTS OF USB  
DEVICE AND METHOD OF CONTROLLING ENDPOINTS OF  
USB DEVICE

**PRIORITY LETTER**

February 27, 2004

Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

Dear Sirs:

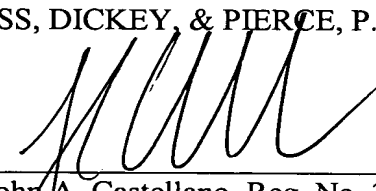
Pursuant to the provisions of 35 U.S.C. 119, enclosed is/are a certified copy of the following priority document(s).

<u>Application No.</u>	<u>Date Filed</u>	<u>Country</u>
10-2003-0057210	August 19, 2003	Korea

In support of Applicant's priority claim, please enter this document into the file.

Respectfully submitted,

HARNESS, DICKEY, & PIERCE, P.L.C.

By   
John A. Castellano, Reg. No. 35,094  
P.O. Box 8910  
Reston, Virginia 20195  
(703) 668-8000

HARNESS, DICKEY & PIERCE, P.C.  
703-668-8000



별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto  
is a true copy from the records of the Korean Intellectual  
Property Office.

출원번호 : 10-2003-0057210  
Application Number

출원년월일 : 2003년 08월 19일  
Date of Application AUG 19, 2003

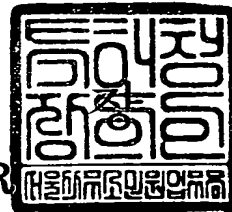
출원인 : 삼성전자주식회사  
Applicant(s) SAMSUNG ELECTRONICS CO., LTD.



2003 년 10 월 27 일

특 허 청

COMMISSIONER





1020030057210

출력 일자: 2003/10/31

**【서지사항】**

<b>【서류명】</b>	특허출원서
<b>【권리구분】</b>	특허
<b>【수신처】</b>	특허청장
<b>【제출일자】</b>	2003.08.19
<b>【발명의 명칭】</b>	유에스비 디바이스의 엔드포인트 제어 장치 및 그 방법
<b>【발명의 영문명칭】</b>	Endpoint Controller of USB device and Method of the same
<b>【출원인】</b>	
<b>【명칭】</b>	삼성전자주식회사
<b>【출원인코드】</b>	1-1998-104271-3
<b>【대리인】</b>	
<b>【성명】</b>	박영우
<b>【대리인코드】</b>	9-1998-000230-2
<b>【포괄위임등록번호】</b>	1999-030203-7
<b>【발명자】</b>	
<b>【성명의 국문표기】</b>	임충빈
<b>【성명의 영문표기】</b>	LIM, Choong Bin
<b>【주민등록번호】</b>	621227-1024922
<b>【우편번호】</b>	449-846
<b>【주소】</b>	경기도 용인시 수지읍 풍덕천리 693 삼성1차아파트 104동 1701호
<b>【국적】</b>	KR
<b>【발명자】</b>	
<b>【성명의 국문표기】</b>	전세훈
<b>【성명의 영문표기】</b>	JEON, Se Hoon
<b>【주민등록번호】</b>	730309-1850910
<b>【우편번호】</b>	445-973
<b>【주소】</b>	경기도 화성시 태안읍 반월리 868 신영통현대아파트 206동503호
<b>【국적】</b>	KR
<b>【심사청구】</b>	청구
<b>【취지】</b>	특허법 제42조의 규정에 의한 출원, 특허법 제60조의 규정에 의한 출원심사를 청구합니다. 대리인 박영우 (인)



1020030057210

출력 일자: 2003/10/31

【수수료】

【기본출원료】 20 면 29,000 원

【가산출원료】 13 면 13,000 원

【우선권주장료】 0 건 0 원

【심사청구료】 10 항 429,000 원

【합계】 471,000 원

【첨부서류】 1. 요약서·명세서(도면)\_1통

**【요약서】****【요약】**

목표 응용이 정해지지 않은 USB단품 디바이스에서 사용자가 정해진 버퍼 사이즈내에서 원하는 엔드포인트의 스펙대로 최대의 효율을 얻게하는 것을 목적으로 하는 본 발명은 USB 2.0 호스트로부터 USB 2.0 디바이스로 데이터를 전달하기 위한 OUT 엔드포인트 버퍼의 구조를 최대 패킷 사이즈\* $L$ 로 정의하고, 각 엔드포인트에 대한 버퍼 사이즈는 사이즈 제어 레지스터 셋을 통해 프로그래머블하게 조정할 수 있게 한 것이다. 이를 위하여 본 발명의 엔드포인트 제어기는 버퍼의 사이즈 제어 레지스터 셋과 다른 레지스터들을 실제 수신상황에 따라 적절하게 셋팅하여 각 엔드포인트에 대한 버퍼 사이즈를 적응적으로 조절한다. 즉, 본 발명의 엔드포인트 제어기는 일정 주기  $T$  동안 각 엔드포인트별로 NAK가 발생된 빈도를 하드웨어적으로 체크하여 문턱치를 벗어나면 인터럽트를 발생하여 상위 레이어가 알 수 있도록 함으로써 필요시 소프트웨어가 각 아웃 엔드포인트의 할당 영역을 수신상황에 맞게 적응적으로 구성할 수 있게 한다.

**【대표도】**

도 11

**【색인어】**

USB, 디바이스, 호스트, FIFO, 버퍼, 엔드포인트, 제어기, 버퍼 사이즈

**【명세서】****【발명의 명칭】**

유에스비 디바이스의 엔드포인트 제어 장치 및 그 방법{Endpoint Controller of USB device and Method of the same }

**【도면의 간단한 설명】**

- 도 1은 USB 규격에 따른 USB 버스 토폴로지를 도시한 도면,
- 도 2는 USB 규격에 따른 USB 호스트와 디바이스의 세부 구조를 도시한 도면,
- 도 3은 USB 규격에 따른 파이프 개념을 설명하기 위해 도시한 도면,
- 도 4는 USB 규격에 따른 버스 트랜잭션의 예를 도시한 도면,
- 도 5는 USB 규격에 따른 패킷 포맷의 예,
- 도 6은 종래 USB 디바이스의 버퍼 구조를 도시한 도면,
- 도 7은 본 발명이 적용되는 USB 디바이스의 구성 블록도,
- 도 8은 본 발명에 따른 엔드포인트 버퍼 구조를 도시한 개략도,
- 도 9는 도 8에 도시된 버퍼의 운영 개념을 도시한 도면,
- 도 10은 본 발명에 따라 폭(Width)이 K 바이트 단위인 버퍼의 예,
- 도 11은 본 발명에 따른 엔드포인트 제어장치를 도시한 구성 블록도,
- 도 12는 도 11에 도시된 FIFO 포인터 제어부의 세부 구성 블록도,
- 도 13은 본 발명에 따른 엔드포인트 제어장치의 동작 절차를 도시한 순서도,



도 14는 본 발명이 적용된 USB 디바이스의 MCU가 버퍼구조를 변경하는 절차를 도시한 순서도.

\*도면의 주요부분에 대한 부호의 설명

710: 직렬 인터페이스 엔진(SIE) 720: 버퍼부

722,722': FIFO 버퍼 724: 버퍼 사이즈 레지스터 셋

726: 최대 패킷 사이즈 레지스터 셋 728: 엔드포인트 버퍼 제어기

730: MCU 인터페이스부 740: MCU

810: FIFO 상태 검사부 820: NAK 카운터부

830: FIFO 포인터 제어부 840: 타이머

850: NAK 문턱치 제어부 832: NAK 카운트 레지스터

834: 비교기 836: 포인터 레지스터 셋

#### 【발명의 상세한 설명】

#### 【발명의 목적】

#### 【발명이 속하는 기술분야 및 그 분야의 종래기술】

<24> 본 발명은 USB(Universal Serial Bus) 디바이스에 사용되는 USB용 반도체 칩에 관한 것으로, 더욱 상세하게는 USB 디바이스의 엔드포인트 제어 장치 및 그 방법에 관한 것이다.

<25> 일반적으로, USB 규격은 호스트(host) 컴퓨터의 시스템 버스에 다양한 주변장치(키보드, 마우스, 모니터, 웹 카메라, 조이스틱, 스토리지 등)를 연결하기 위한 산업규격으로서, 현재 USB 2.0 규격이 2000년 4월 27일 발표되어 사용되고 있



다. USB는 컴퓨터 주변장치에 대해 플러그 앤 플레이를 가능하게 하므로 PC에 주변장치를 연결할 경우 자동으로 인스톨되어 사용을 편리하게 하고, 연결을 간편하게 한다.

<26> USB 시스템은 USB 호스트(host)와 USB 디바이스(device), 및 이들을 연결하는 USB 접속 수단(interconnect)으로 이루어지는데, USB 호스트와 USB 디바이스를 물리적으로 연결하는 버스 토폴로지는 도 1에 도시된 바와 같이 계층화된 스타 토폴로지(tiered star topology)이다. 스타 토폴로지에서 허브(hub)는 디바이스들이 연결되는 중심이 되는데, 호스트(102)는 제1 계층으로서 루트(root) 허브이고, 호스트(102)와 디바이스들은 점 대 점(point-to-point)으로 접속된다. 도 1에서 호스트(102)에 연결된 제1 허브(106)와 평선(104)은 제2 계층이 되고, 제1 허브(106)에 연결된 제2 허브(112)와 평선들(108,110)은 제3 계층이 되며, 동일한 방식으로 최대 제7 계층까지 연결이 가능하다.

<27> USB 호스트(102)는 USB 시스템에서 오직 하나뿐이며 호스트 제어기(host controller)를 포함하고 있고, USB 디바이스(104~116)는 부가적인 접속점을 제공할 수 있는 허브(hub: 106,112)와 조이스틱이나 스피커 등과 같이 시스템에 실제적인 기능을 제공하는 평선(function: 104,108,110,116)으로 이루어지고, 고유의 USB 주소(address)에 의해 액세스될 수 있다. USB 호스트(102)와 디바이스를 연결하는 물리적인 와이어 케이블은 +5V 전원을 제공하는 V<sub>BUS</sub>와 D+, D-의 데이터 선, 접지선(GND) 등 4선으로 되어 있고, 데이터 전송속도는 고속모드(high-speed signaling)에서 480 Mb/s, 전속모드(full-speed signaling)에서 12 Mb/s, 저속모드(low-speed signaling)에서 1.5 Mb/s이다.

<28> USB 호스트와 USB 디바이스의 구체적인 구조는 도 2에 도시된 바와 같다. 도 2를 참조하면, USB 호스트(210)는 호스트 제어기(212a)와 SIE가 있는 USB 버스 인터페이스부(212)와 USB 시스템 소프트웨어(214), 클라이언트 소프트웨어(216)로 계층화된 구조로 되어 있고, USB 디바





이스(220)는 SIE를 포함하는 USB 버스 인터페이스부(222)와 엔드포인트0(EP0)를 포함하는 USB 논리 디바이스(224), 펌션(function: 226)으로 이루어져 호스트(210)에 대응하는 계층구조를 이루고 있다. USB 호스트(210)와 디바이스(220)는 USB 인터페이스부(212, 222)의 SIE를 통해 USB 케이블로 물리적으로 연결되고, USB 시스템 소프트웨어(214)는 엔드포인트0(EP0)로 설정된 디폴트 제어 파이프(Default Control Pipe)를 통해 디바이스(220)를 관리한다. 그리고 클라이언트 소프트웨어(216)와 펌션(226) 사이에는 다수의 파이프가 형성되어 서로 통신하도록 되어 있다. 즉, USB 규격에 따르면 발신지(source)와 착신지(destination) 사이의 USB 데이터 전송은 도 3에 도시된 바와 같이, 호스트(Host)와 디바이스(USB Device)의 엔드포인트(endpoint)를 연결하는 독립적인 채널(이를 '파이프'라 한다)을 통해 이루어진다. 도 3을 참조하면, USB 디바이스는 엔드포인트 번호로 식별되는 다수의 엔드포인트(Endpoints)들로 이루어지고, 호스트 측의 버퍼 메모리(Buffers)와 디바이스의 엔드포인트(Endpoints)는 서로 독립적인 파이프(Pipes)로 연결되어 해당 파이프를 통해 데이터 흐름(Data Flows)이 이루어진다. 각 엔드포인트는 버스 액세스 주파수, 대역폭, 엔드포인트 번호, 에러처리 절차, 엔드포인트가 보내거나 수신할 수 있는 최대 패킷 사이즈, 엔드포인트의 전송 형태, 데이터 전송 방향 등에 의해 그 특성이 정의된다. 그리고 모든 USB 디바이스에서 "엔드포인트0(EP0)"은 디폴트 제어 파이프로서 USB 시스템 소프트웨어가 해당 디바이스를 구성할 때 사용하며, 다른 파이프들은 디폴트 제어 파이프를 통해 디바이스의 구성이 완료되면 형성된다. USB 파이프는 스트림(stream) 타입과 메시지(message) 타입으로 구분되고, 전송 형태(transfer type)는 제어 전송(control transfer), 동기전송(Isochronous transfer), 인터럽트(Interrupt) 전송, 벌크(Bulk) 전송이 있다.

<29> 한편, USB 버스 프로토콜은 폴드 토큰 버스(polled token bus)로서 호스트 제어기가 모든 데이터 전송을 관장한다. 대부분의 버스 트랜잭션(bus transaction)은 도 4에 도시된 바와 같이, 3 패킷의 전송으로 이루어지는데, 호스트 제어기가 트랜잭션의 타입과 방향, USB 디바이스 주소, 엔드포인트 번호가 실린 패킷(이를 '토큰 패킷'이라 한다)을 보냄으로써 트랜잭션이 시작된다. 그리고 트랜잭션에서 데이터 전송은 (A)와 같이 디바이스로부터 호스트 방향(이를 "IN" 방향이라 한다)으로, 혹은 (B)와 같이 호스트에서 디바이스 방향(이를 "OUT" 방향이라 한다)으로 이루어지는데, 데이터를 수신한 호스트나 디바이스는 핸드셰이크 패킷으로 응답한다. 이때 "ACK" 핸드셰이크 패킷은 정상적으로 데이터를 수신한 것을 나타내고, "NACK" 핸드셰이크 패킷은 수신시 에러가 발생되거나 수신하지 못함을 나타내며, NACK를 수신한 상대방은 데이터를 재전송해야 한다.

<30> 각 패킷은 패킷 식별자(PID: Packet Identifier)에 의해 다음 표 1과 같이 구분되고, 각 패킷의 대표적인 포맷은 도 5에 도시된 바와 같다.

<31> 【표 1】

PID 타입	PID 명칭	PID<3:0>	설명
Token	OUT	0001B	아웃방향 데이터 전송을 위한 토큰
	IN	1001B	인방향 데이터 전송을 위한 토큰
	SOF	0101B	프레임 시작표시(1ms 혹은 125 $\mu$ s)
	SETUP	1101B	셋업
Data	DATA0	0011B	데이터 패킷 PID 짝수
	DATA1	1011B	데이터 패킷 PID 홀수
	DATA2	0111B	데이터 패킷 고속모드
	MDATA	1111B	데이터 패킷 고속모드
Handshake	ACK	0010B	에러없이 데이터 수신 응답
	NAK	1010B	에러발생 수신 응답
	STALL	1110B	컨트롤 파이프 준비안됨
	NYET	0110B	응답없음
Special	PRE	1100B	
	ERR	1100B	
	SPLIT	1000B	
	PING	0100B	

- <32> 도 5를 참조하면, (A)는 토큰 패킷의 포맷을, (B)는 SOF(Start-Of-Frame) 패킷의 포맷을, (C)는 데이터 패킷의 포맷을, (D)는 핸드셰이크 패킷의 포맷을 나타낸다.
- <33> 토큰 패킷(A)은 8비트의 PID와 7비트의 어드레스(ADDR), 4비트의 엔드포인트 번호(ENDP), 5비트의 CRC 등 총 3바이트로 이루어지고, SOF 패킷은 8비트의 PID와 11비트의 프레임 번호, 5비트의 CRC5로 이루어진다. SOF 패킷은 프레임의 시작을 알려주는데, 전속(Full-speed) 모드의 경우 프레임 주기는 1ms이고, 고속(High-speed)모드는 125 $\mu$ s이다. 데이터 패킷(C)은 8비트의 PID와 0~8192비트의 데이터, 16비트의 CRC로 이루어지며, 핸드셰이크 패킷(D)은 8비트의 PID로만 이루어진다. 따라서 USB 비다이스는 어드레스가 7비트이므로 0~127까지 최대 128개의 주소를 할당할 수 있는 데, 호스트를 고려하면 최대 127개의 디바이스를 연결할 수 있고, 각 디바이스당 엔드포인트는 최대 16개(4비트 번호)를 할당할 수 있다.
- <34> 이와 같은 디바이스 구조에서 호스트와 디바이스 간에 이루어지는 OUT 방향 데이터 전송의 성능은 각 엔드포인트에 사용되는 버퍼의 구조에 따라 크게 좌우된다. 즉, 호스트로부터 데이터를 수신하기 위한 엔드포인트의 버퍼가 하나일 경우 버퍼에 데이터가 차면, NAK이 발생되어 전송이 일시 중지되므로 전송속도가 저하되는 문제점이 있다. 이러한 문제점을 해결하기 위해 종래에 널리 사용되는 아웃 엔드포인트 버퍼는 도 6에 도시된 바와 같이, 호스트(610)와 연결되는 USB 디바이스(620)의 SIE(622)와 MCU 인터페이스(626) 사이에 각 엔드포인트(EP0~EPx)당 2개의 버퍼(a,b)가 할당되는 평풍 버퍼(624) 구조이다.
- <35> 도 6을 참조하면, 평풍 버퍼(624)는 하나의 엔드포인트당 2개의 버퍼(a,b)를 구비하여 어떤 엔드포인트가 호스트로부터 데이터를 받아 a 버퍼에 저장해 놓고 MCU가 이를 가져가기 전에 다시 동일한 엔드포인트로 데이터가 수신되면, 나중에 수신되는 데이터는 b 버퍼에 저장하도록 하여 NAK이 발생되지 않도록 하는 구조이다. 이후 MCU가 a 버퍼의 데이터를 가져가면 다

음에 수신되는 데이터는 다시 a 버퍼에 저장한다. 이와 같이 펌프 버퍼는 2개의 버퍼로 서로 번갈아가면서 데이터를 수신하는 것이다.

<36> 그런데 USB 디바이스의 엔드포인트에 펌프 버퍼를 적용할 경우 각 엔드포인트당 "최대 패킷 사이즈 \* 2" 개의 버퍼를 준비하고 있어야 하므로 USB 규격에서 정의한 최대 16개의 엔드포인트를 사용하게 되면, "최대 패킷 사이즈 \* 2 \* 16" 개의 버퍼가 필요하게 되므로 버퍼의 사이즈가 과도하게 늘어나 칩의 영역이 커지고 전력을 많이 소비되며 비용이 비싸지는 문제점이 있다.

<37> 더욱이 2개의 버퍼를 준비해 놓았다 하더라도 해당 엔드포인트에 할당된 응용(application)의 데이터 교환량이 다른 엔드포인트에 비해 많아 일시에 데이터량이 급증할 경우 2개의 버퍼로도 감당하지 못해 NAK이 자주 발생하게 되고, 상대적으로 통신량(Throughput)이 적은 엔드포인트의 경우에는 하나의 버퍼만으로도 충분하여 디바이스 전체의 자원이 비효율적으로 사용되는 문제점이 있다.

#### 【발명이 이루고자 하는 기술적 과제】

<38> 본 발명은 상기와 같은 문제점을 해결하기 위해 제안된 것으로, 전체 엔드포인트 버퍼의 구조를 최대 패킷 사이즈\*L로 두고, 각 엔드포인트당 버퍼의 수를 데이터 전송빈도에 따라 적응적으로 설정할 수 있게 함으로써 통신량이 많은 엔드포인트는 보다 많은 수의 버퍼를 갖게 하고, 통신량이 적은 엔드포인트는 적은 수의 버퍼를 갖게 하여 데이터를 수신하는 동안 NAK의

발생을 최소화하여 전체적으로 성능이 개선된 USB 디바이스의 엔드포인트 제어장치 및 제어방법을 제공하는데 그 목적이 있다.

### 【발명의 구성 및 작용】

- <39>       상기와 같은 목적을 달성하기 위하여 본 발명의 장치는 USB 규격에 따라 직렬 인터페이스 엔진(SIE)을 통해 호스트와 접속되고 주변장치 프로세서(MCU)에 의해 기능(Function)을 수행하도록 된 디바이스에 있어서, 각 엔드포인트별로 할당되는 선입선출(FIFO) 버퍼와; 상기 직렬 인터페이스 엔진을 통해 호스트와 패킷을 송수신하면서 일정주기로 각 엔드포인트별로 수신 실패율을 카운트하여, 수신 실패가 최소가 되도록 각 엔드포인트별로 상기 선입선출(FIFO) 버퍼를 재설정하도록 제어하는 엔드포인트 버퍼 제어기를 포함하는 것을 특징으로 한다.
- <40>       이때, 상기 선입선출 버퍼는 최대 패킷 사이즈\* L의 크기로 설정되고, 상기 엔드포인트 버퍼 제어기는 버퍼상태 정보를 토대로 패킷을 수신할 수 있는지 없는지를 결정하며, 수신불가 시에는 NAK 핸드셰이트 패킷을 발생시키는 버퍼상태 검사부; 일정주기 T 주기마다 NAK 카운팅 리셋신호를 발생시키는 타이머; 상기 T 주기내에 각 엔드포인트별로 발생된 NAK을 카운팅하고, 상기 타이머로부터 리셋신호가 발생되면 각 엔드포인트별 NAK 카운트값을 넘겨주는 NAK 카운터; 및 상기 NAK 카운터로부터 수신된 각 엔드포인트별 NAK 카운트값을 토대로 초기에 정해진 NAK 문턱치를 넘는 엔드포인트가 검출되면 상기 MCU로 인터럽트를 발생시키는 FIFO 포인터 제어부를 구비한다.
- <41>       상기와 같은 목적을 달성하기 위하여 본 발명의 방법은, 버퍼를 각 엔드포인트별로 디폴트로 할당하는 단계; 각 엔드포인트별로 데이터를 수신하면서 NAK이 발생되면 이를 카운트

하는 단계; 일정주기가 되면 상기 NAK 카운트값을 미리 설정된 문턱치와 비교하는 단계; 및 비교결과 문턱치를 벗어나면, NAK 카운트값에 따라 버퍼를 재설정하는 단계를 구비한 것을 특징으로 한다.

<42> 이하, 첨부된 도면을 참조하여 본 발명의 바람직한 실시예를 자세히 설명하기로 한다.

<43> 본 발명이 적용되는 USB 디바이스(700)는 도 7에 도시된 바와 같이, 크게 직렬 인터페이스 엔진(Serial Interface Engine: SIE; 710)과, 버퍼부(720)와, MCU인터페이스부(730), 주변장치 프로세서(MCU: 740)로 구성되어 USB 규격에 따라 호스트와 접속된다.

<44> 도 7을 참조하면, 직렬 인터페이스 엔진(710)은 디바이스(700)를 USB 프로토콜에 따라 허브나 호스트에 연결하기 위한 부분이고, 버퍼부(720)는 호스트로부터 수신된 데이터나 호스트로 전송할 데이터를 일시 저장하는 부분이며, 주변장치 프로세서(MCU:740)는 본 발명에 따라 버퍼의 사이즈를 재설정하는 기능과 디바이스 고유의 기능(Function)을 실행한다. 여기서, 버퍼부(720)는 호스트로부터 디바이스로 데이터를 수신하기 위한 아웃(OUT) 선입선출(FIFO) 버퍼(도9의 722)와, 디바이스로부터 호스트로 전송할 데이터를 일시 저장하는 인(IN) 선입선출(FIFO) 버퍼가 있을 수 있는데, 이들 FIFO 버퍼는 엔드포인트 버퍼 제어기(도 9의 728)에 의해 제어되며, 통상 버퍼의 효율적인 활용은 USB 시스템의 성능과 직결된다.

<45> 도 8은 본 발명에 따른 엔드포인트 버퍼 구조를 도시한 개략도이다.

<46> 도 8을 참조하면, 본 발명에 따른 엔드포인트 FIFO 버퍼(722)는 전체 엔드포인트들을 위한 버퍼를 최대 패킷 사이즈\*L 크기의 통합 버퍼(이를 '다중 패킷 버퍼'라고도 한다)로 만들고, 각 엔드포인트(응용)당 필요한 최대 패킷 사이즈를 정의하는 최대 패킷 사이즈 레지스

터 셋(726)과, 각 엔드포인트당 필요한 유닛의 개수를 결정하는 버퍼 사이즈 레지스터 셋(724)으로 정의된다. 최대 패킷 사이즈 레지스터(726)는 각 엔드포인트당 필요한 최대 패킷의 크기를 정해주고, 이 값은 해당 엔드포인트가 가질 멀티플 버퍼(Multiple Buffer)의 한 단위(Unit)가 되며, 이 값은 소프트웨어적으로 정해진다. 버퍼 사이즈 레지스터(724)에는 각 엔드포인트에 할당할 유닛의 갯수가 정의되고, 전체 레지스터 값의 합이 L을 넘지 않도록 하드웨어적으로 보호수단을 준비한다(즉, Protection을 걸어준다). 통상, 이 두 레지스터는 엔드포인트 제어기 내부에 위치하는 것으로, 프로그램이나 하드웨어에 의해 액세스될 수 있다. 본 발명의 실시예에서는 버퍼 사이즈 레지스터 셋(724)과 최대 패킷 사이즈 레지스터 셋(726)이 엔드포인트 버퍼 제어기의 FIFO 포인터 제어기에 위치하는 것으로 한다.

<47> 이러한 다중 패킷 버퍼(722)의 운용은 도 9에 도시된 바와 같다. 도 9를 참조하면, 리셋 후 초기상태에서 각 엔드포인트는 모두 같은 크기의 영역 혹은 프로그래머에 의해 엔드포인트(EP0~EPx)당 필요한 예상치의 크기로 초기화된다. 도 9에서 EP0은 M개의 유닛이 할당되어 있고, EPx는 N개의 유닛이 할당되어 있다.

<48> 따라서 수신된 패킷은 해당 엔드포인트에 할당된 다중 버퍼중 제일 첫번째 최대 패킷 유닛부터 차례대로 채워지고, MCU도 첫번째 유닛부터 차례대로 데이터를 읽어가 서비스한다. 그 다음부터는 리셋이 되지 않는 한 링(Ring)의 형태로 호스트로부터의 데이터 수신과 MCU에 의한 데이터 서비스가 반복된다. 이때 최대 패킷 사이즈가 각 엔드포인트마다 서로 다를 수가 있으므로 공간효율을 최대화할 수 있도록 도 10에 도시된 바와 같이, 폭(Width)이 K 바이트 단위인 버퍼(722')를 사용하는 것이 바람직하다(여기서,  $K=1, 2, \dots, K$ ). 도 10을 참조하면, 버퍼(722')의 폭

은 K 바이트이고, 전체 길이는 L이며, 엔드포인트0는 정해진 최대 패킷 사이즈의 버퍼를 M개 가지고 있고, 엔드포인트x는 정해진 최대 패킷 사이즈의 버퍼를 N개 가지고 있다. 그리고 엔드포인트0의 한 유닛은 엔드포인트0의 최대 패킷 사이즈를 K 바이트로 나눈 수의 행으로 이루어지고, 엔드포인트 x의 한 유닛은 엔드포인트 x의 최대 패킷 사이즈를 K 바이트로 나눈 수의 행으로 이루어진다.

<49> 도 11은 본 발명에 따른 엔드포인트 제어장치를 도시한 구성 블록도이다.

<50> 도 11을 참조하면, 본 발명의 엔드포인트 제어장치는 엔드포인트 버퍼 제어기(728)와 엔드포인트 FIFO버퍼(722)로 이루어지고, 엔드포인트 버퍼 제어기(728)는 선입선출 버퍼 상태 검사부(810)와, NAK 카운터(820), 타이머(840), FIFO 버퍼 포인터 제어부(830), NAK 문턱치(Threshold) 제어부(850)로 구성되어 해당 엔드포인트 FIFO 버퍼(722)가 가득(Full)차면 SIE(710)에 NACK를 발생하고, NAK카운트값이 문턱치(Threshold)를 넘어서면 MCU(740)에 인터럽트를 발생하며, 상위 소프트웨어의 제어에 따라 엔드포인트 FIFO 버퍼(722)의 포인터를 설정한다.

<51> FIFO 버퍼상태 검사부(810)는 FIFO 버퍼 포인터 제어부(830)로부터 오는 버퍼상태 정보(FULL, HALF, EMPTY 등 버퍼의 충만도를 나타내는 정보)를 토대로 호스트로부터 수신되는 아웃패킷을 수신할 수 있는지 없는지를 결정하며, 수신불가시에는 NAK 핸드셰이트 패킷을 발생시킨다.

<52> NAK 카운터(820)는 타이머(840)에 정의된 일정 주기의 T 주기내에 각 엔드포인트별로 발생된 NAK을 카운팅하여 타이머(840)에서 리셋신호가 발생되면 FIFO 포인터 제어부(830)로 각 엔드포인트별 NAK 카운트값을 넘겨준다. 이때 NAK카운팅은 NAK N개당 1씩 증가하는 형태로 이루어지는 것이 바람직하다(여기서,  $N=1, 2, \dots, N$ ).



- <53> 타이머(840)는 프로그래머가 소프트웨어적으로 정할 수 있는 일정주기 T 주기마다 NAK 카운팅 리셋신호를 발생시킨다. 시간의 측정(카운트)은 실시간으로 카운터를 돌리면 하드웨어적으로 무리가 가고 전력소모가 크므로 USB 프로토콜상 전속(Full-speed)모드의 프레임 주기인 1ms마다 호스트로부터 전달되는 SOF를 카운팅하는 것이 바람직하다.
- <54> 문턱치 제어부(850)는 프로그래머가 설정할 수 있는, 패킷 수신시 발생하는 NAK 개수의 문턱치(Threshold)를 설정한다. 즉, 문턱치 제어부(850)는 발생하는 NAK의 적정 수준을 문턱치로서 정의하고, 각 엔드포인트당 발생하는 NAK의 개수가 문턱치를 넘으면 FIFO 포인터 제어부(830)가 MCU(840)쪽으로 인터럽트를 발생시켜 소프트웨어가 해당 엔드포인트에 대해 사이즈를 늘리는 조치를 취할 수 있게 한다.
- <55> FIFO 포인터 제어부(830)는 도 12에 도시된 바와 같이, NAK 카운트 레지스터 셋(832)과 비교기(834)로 이루어져 카운터로부터 수신된 각 엔드포인트별 NAK 카운트값을 토대로 초기에 정해진 NAK 문턱치(850)를 넘는 엔드포인트가 검출되면 MCU(840)로 인터럽트를 발생시킨다. 이때 MCU(840)는 인터럽트를 수신하면 후술하는 바와 같이 NAK 카운트 레지스터를 검사한 후 NAK의 빈도를 근거로 각 엔드포인트 FIFO 버퍼의 영역을 재설정한다. 영역 재설정시에는 현재 엔드포인트 FIFO 버퍼(722)에 수신된 데이터는 모두 MCU(840)에 의해 읽혀진 상태이어야 하고, 재설정중에 호스트로부터 패킷이 수신되면 해당 엔드포인트에 대해 NAK을 발생시켜 패킷 수신을 잠시 차단시켜야 한다. 또한 FIFO 포인터 제어부(830)는 각 엔드포인트별로 버퍼 영역을 할당하기 위한 버퍼 사이즈 레지스터(724)와 최대 패킷 사이즈 레지스터(726), 및 포인터 레지스터(836)를 구비하고 있다.
- <56> 도 13은 본 발명에 따라 엔드포인트 버퍼 제어기를 운용하는 절차를 도시한 순서도이다.

- <57> 리셋이 발생되거나 초기전원이 공급되면 MCU에 의해 각 레지스터를 초기화한다(S1).  
이때 엔드포인트별 버퍼 사이즈를 구성하고, NAK 문턱치를 설정하며, 주기 T를 설정한다.
- <58> 이어 각 엔드포인트별로 호스트로부터 데이터 수신동작이 일어나면, 엔드포인트 버퍼 제어기(728)가 초기 설정값으로 해당 아웃 엔드포인트 버퍼(722)를 사용하여 패킷을 수신하고, 버퍼(722)가 차면 버퍼상태 제어부(810)가 NAK을 발생시킴과 아울러 NAK 카운터(820)에서 NAK을 카운트한다(S2).
- <59> 이어 매 T 주기마다 NAK 카운터(820)의 카운트값을 포인터 제어부(830)의 NAK 카운트 레지스터(832)로 전달하고, 비교기(834)는 이를 설정된 문턱치(850)와 비교한다(S3-S5).
- <60> 비교결과, 각 엔드포인트의 NAK 카운트값이 문턱치 이하이면, NAK 카운터 및 타이머를 리셋한 후 데이터 수신동작을 반복한다(S6,S7).
- <61> 만일, 비교결과 NAK 카운트값이 문턱치를 넘어가면 MCU 인터럽트를 발생시켜 MCU에 의해 재설정 이 이루어지게 함과 아울러, 재설정 과정에서 수신되는 아웃 패킷에 대해 NAK으로 대응하고, FIFO에 이미 저장된 데이터를 처리하는 동작(FIFO Flushing)을 수행한다(S8,S9).
- <62> 도 14는 본 발명이 적용된 USB 디바이스의 MCU가 버퍼를 재설정하는 절차를 도시한 순서도이다.
- <63> 도 14를 참조하면, MCU는 인터럽트가 발생되면 버퍼 재설정 소프트웨어를 구동한다(S11,S12).
- <64> 재설정 소프트웨어는 엔드포인트 버퍼 제어기의 최대 패킷 사이즈 레지스터(726)와 버퍼 사이즈 레지스터(724)의 값을 읽어와 각 엔드포인트별 버퍼의 현재 설정상태를 파악하고, FIFO

포인터 제어기의 NAK 카운트 레지스터(832)로부터 각 엔드포인트별 NAK 카운트값을 읽어온다(S13,S14).

<65> 이어 소정의 버퍼 할당 알고리즘에 따라 각 엔드포인트별로 버퍼를 재할당하고, 이에 따라 엔드포인트 버퍼 제어기(728)의 각 엔드포인트별 FIFO 사이즈를 재구성한다(S15,S16).

<66> 이때 버퍼 할당 알고리즘으로는 매우 다양한 방식이 가능하다. 예컨대, NAK 발생 카운트값이 가장 적은 엔드포인트 버퍼의 버퍼 사이즈를 줄이고, NAK 발생 카운트값이 가장 큰 엔드포인트의 버퍼 사이즈를 증가시키거나 NAK 발생 카운트값에 비례하여 전체 버퍼를 각 엔드포인트별로 재할당시킬 수도 있다. 또한 소수의 버퍼영역을 자유영역으로 유보시켜 놓은 후, 인터럽트를 발생시킨 엔드포인트의 버퍼에 자유영역의 일부를 추가로 할당시킬 수도 있다.

#### 【발명의 효과】

<67> 이상에서 설명한 바와 같이, 본 발명에 따르면 USB 호스트로부터 디바이스의 한 엔드포인트로 대량의 데이터가 수신되면 해당 엔드포인트에 할당된 여러개의 최대 패킷 사이즈의 버퍼가 순차적으로 수신되는 데이터를 받아내므로 연속적으로 들어오는 아웃 패킷 데이터를 NAK 없이 수신할 수 있다. 그리고 초기에 예상하여 설정했던 각 엔드포인트당 버퍼의 개수를 실제 수신시의 실패율(NAK 카운트값)을 측정하여 소프트웨어가 적응적으로 FIFO 버퍼 크기를 재설정할 수 있으므로 자원을 효율적으로 사용할 수 있고 수신율을 향상시킬 수 있는 장점이 있다.

<68>       상기에서는 본 발명의 바람직한 실시예를 참조하여 설명하였지만, 해당 기술 분야의 숙련된 당업자는 하기의 특허 청구의 범위에 기재된 본 발명의 사상 및 영역으로부터 벗어나지 않는 범위내에서 본 발명을 다양하게 수정 및 변경시킬 수 있음을 이해할 수 있을 것이다.

**【특허청구범위】****【청구항 1】**

USB 규격에 따라 직렬 인터페이스 엔진(SIE)을 통해 호스트와 접속되고 프로세서(MCU)에 의해 기능(Function)을 수행하도록 된 디바이스에 있어서,

각 엔드 포인트 별로 할당되는 선입선출(FIFO) 버퍼와;

상기 직렬 인터페이스 엔진을 통해 호스트와 패킷을 송수신하면서 일정주기로 각 엔드 포인트 별로 수신 실패율을 카운트하여, 수신 실패가 최소가 되도록 각 엔드 포인트 별로 상기 선입선출(FIFO) 버퍼를 재설정하도록 제어하는 엔드 포인트 버퍼 제어기를 포함하는 것을 특징으로 하는 USB 디바이스의 엔드 포인트 제어장치.

**【청구항 2】**

제1항에 있어서, 상기 선입선출 버퍼는 최대 패킷 사이즈  $N$ 의 크기로 설정된 것을 특징으로 하는 USB 디바이스의 엔드포인트 제어장치.

**【청구항 3】**

제1항에 있어서, 상기 엔드포인트 버퍼 제어기는

버퍼상태 정보를 토대로 호스트로부터 수신되는 아웃 패킷을 수신할 수 있는지 없는지를 결정하며, 수신 불가 시에는 NAK 핸드셰이트 패킷을 발생시키는 버퍼상태 검사부;

일정주기  $T$  주기마다 NAK 카운팅 리셋신호를 발생시키는 타이머;

상기  $T$  주기내에 각 엔드 포인트 별로 발생된 NAK을 카운팅하고, 상기 타이머로부터 리셋신호가 발생되면 각 엔드포인트별 NAK 카운트값을 넘겨주는 NAK 카운터; 및

상기 NAK 카운터로부터 수신된 각 엔드포인트별 NAK 카운트값을 토대로 초기에 정해진 NAK 문턱치를 넘는 엔드포인트가 검출되면 상기 MCU로 인터럽트를 발생시키는 FIFO 포인터 제어부를 구비하는 것을 특징으로 하는 USB 디바이스의 엔드포인트 제어장치.

#### 【청구항 4】

제3항에 있어서, 상기 엔드포인트 제어장치는 패킷 수신시 발생하는 NAK 개수의 문턱치를 설정하는 문턱치 제어부를 더 구비하는 것을 특징으로 하는 USB 디바이스의 엔드포인트 제어장치.

#### 【청구항 5】

제4항에 있어서, 상기 주변장치 프로세서(MCU)는

상기 인터럽트가 발생되면, 소정의 알고리즘에 따라 상기 엔드포인트 버퍼 제어기의 레지스터를 재설정하여 각 엔드포인트당 버퍼의 개수를 실제 수신시의 실패율(NAK 카운트값)에 따라 적응적으로 재설정하는 것을 특징으로 하는 USB 디바이스의 엔드포인트 제어장치.

#### 【청구항 6】

제3항에 있어서, 상기 타이머는

USB 프로토콜상 전속(Full-speed)모드의 프레임 주기인 1ms마다 호스트로부터 전달되는 SOF를 카운팅하여 주기를 제공하는 것을 특징으로 하는 USB 디바이스의 엔드포인트 제어장치.

#### 【청구항 7】

제3항 또는 제4항에 있어서, 상기 FIFO 포인터 제어부는

엔드포인트별로 NAK카운트값을 저장하기 위한 NAK 카운트 레지스터 셋과, 상기 레지스터 셋의 NAK 카운트값과 상기 문턱치를 비교하는 비교기와, 엔드포인트별 최대 패킷 사이즈를 설

정하기 위한 최대 패킷 사이즈 레지스터 셋과, 엔드포인트별 사이즈를 설정하기 위한 버퍼 사이즈 레지스터 셋을 구비한 것을 특징으로 하는 USB 디바이스의 엔드포인트 제어장치.

**【청구항 8】**

버퍼를 각 엔드포인트별로 디폴트로 할당하는 단계;

각 엔드포인트별로 데이터를 수신하면서 NAK이 발생되면 이를 카운트하는 단계;

일정주기가 되면 상기 NAK 카운트값을 미리 설정된 문턱치와 비교하는 단계; 및

비교결과 문턱치를 벗어나면, NAK 카운트값에 따라 버퍼를 재설정하는 단계를 구비한 것을 특징으로 하는 USB 디바이스의 엔드포인트 제어 방법.

**【청구항 9】**

제8항에 있어서, 상기 일정주기는

USB 프로토콜상 전속(Full-speed)모드의 프레임 주기인 1ms마다 호스트로부터 전달되는 SOF를 카운팅하여 제공하는 것을 특징으로 하는 USB 디바이스의 엔드포인트 제어 방법.

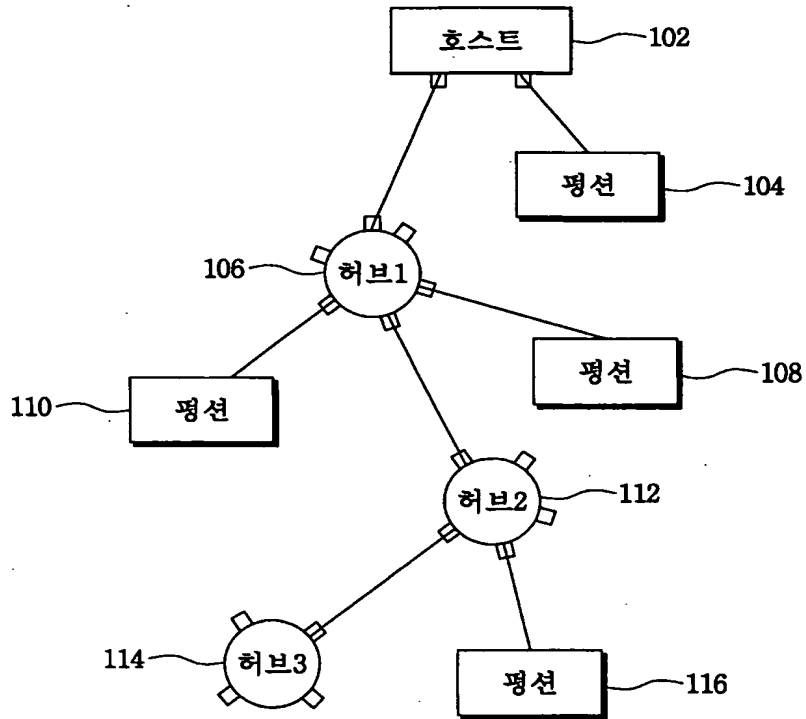
**【청구항 10】**

제8항에 있어서, 상기 재설정하는 단계는

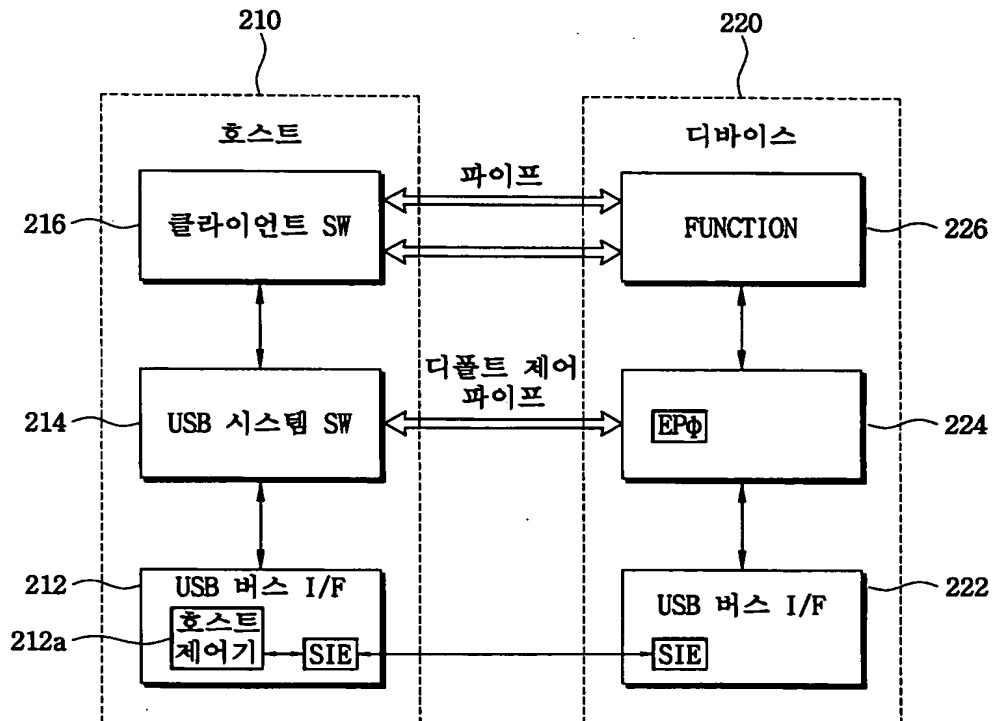
상기 비교결과 인터럽트가 발생되면 주변장치 프로세서가 각 엔드포인트별 NAK 카운트값에 따라 재설정하는 것을 특징으로 하는 USB 디바이스의 엔드포인트 제어 방법.

【도면】

【도 1】

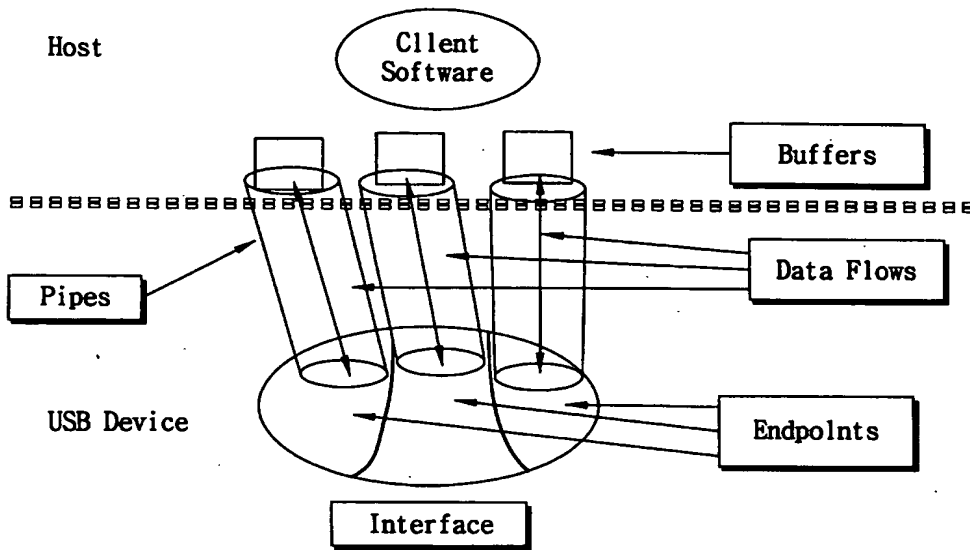


【도 2】

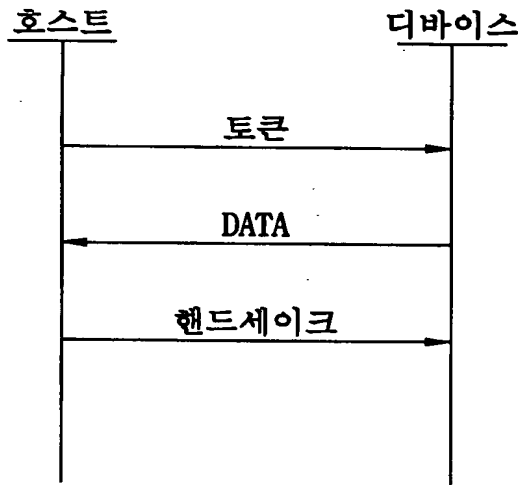




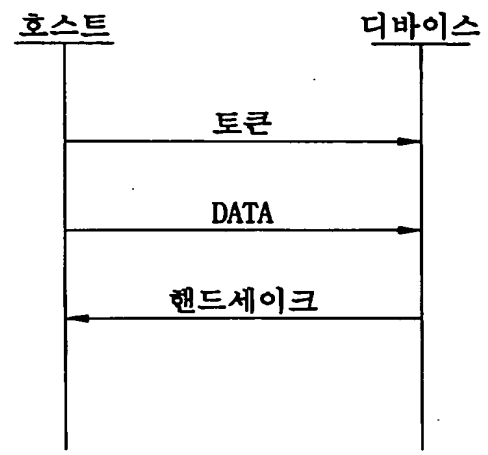
【도 3】



【도 4a】



【도 4b】



【도 5a】

필드	PID	ADDR	ENDP	CRC5
비트	8	7	4	5

【도 5b】

필드	PID	프레임 번호	CRC5
비트	8	11	5

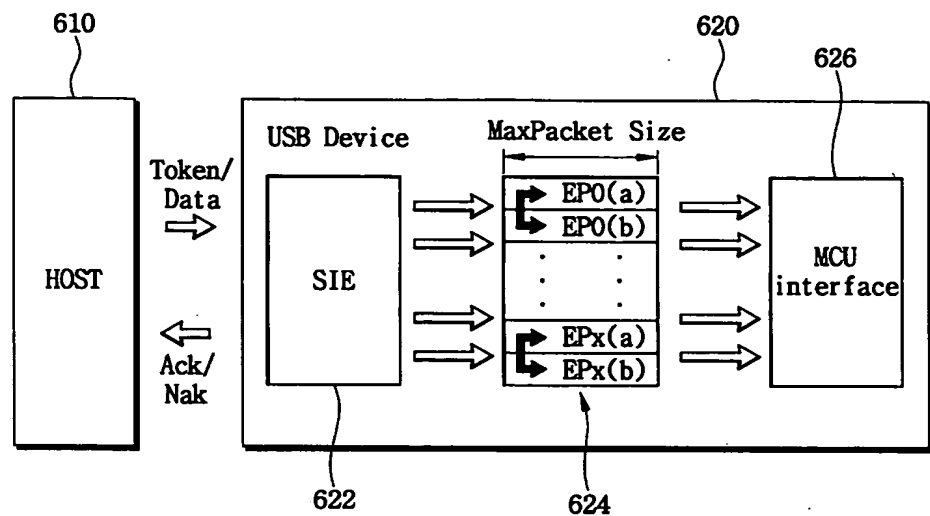
【도 5c】

필드	PID	DATA	CRC16
비트	8	0-8192	16

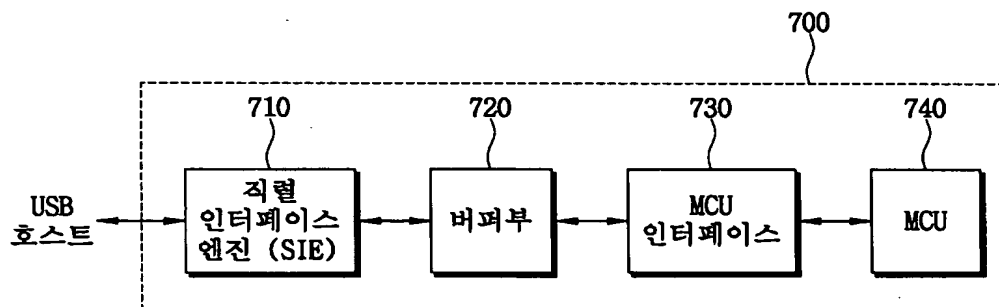
【도 5d】

필드	PID
비트	8

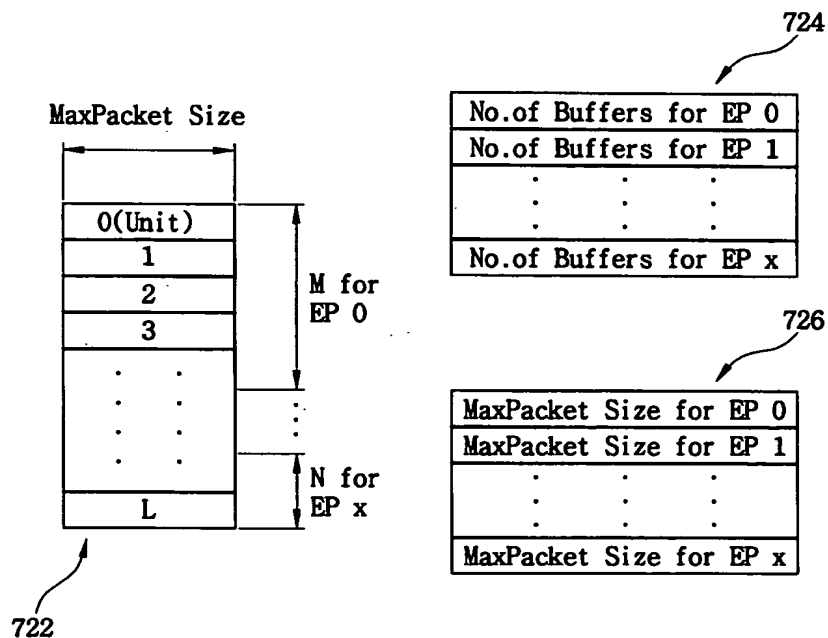
【도 6】



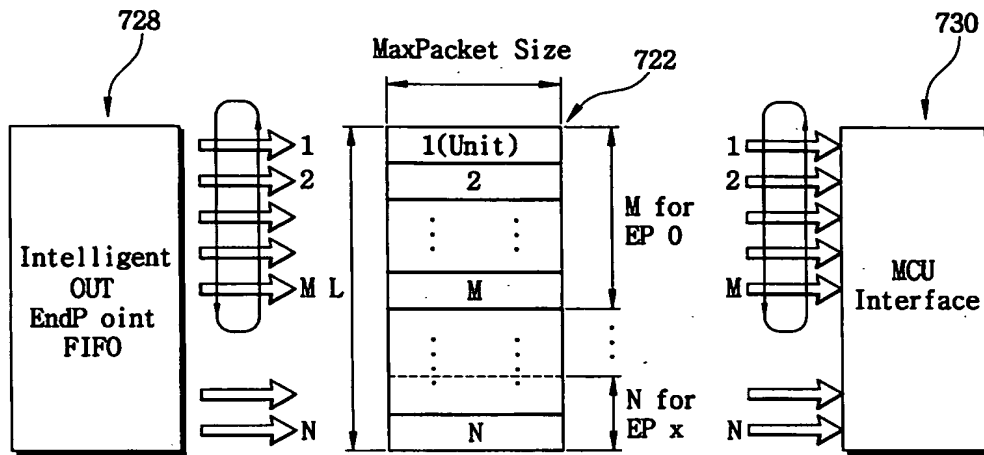
【도 7】



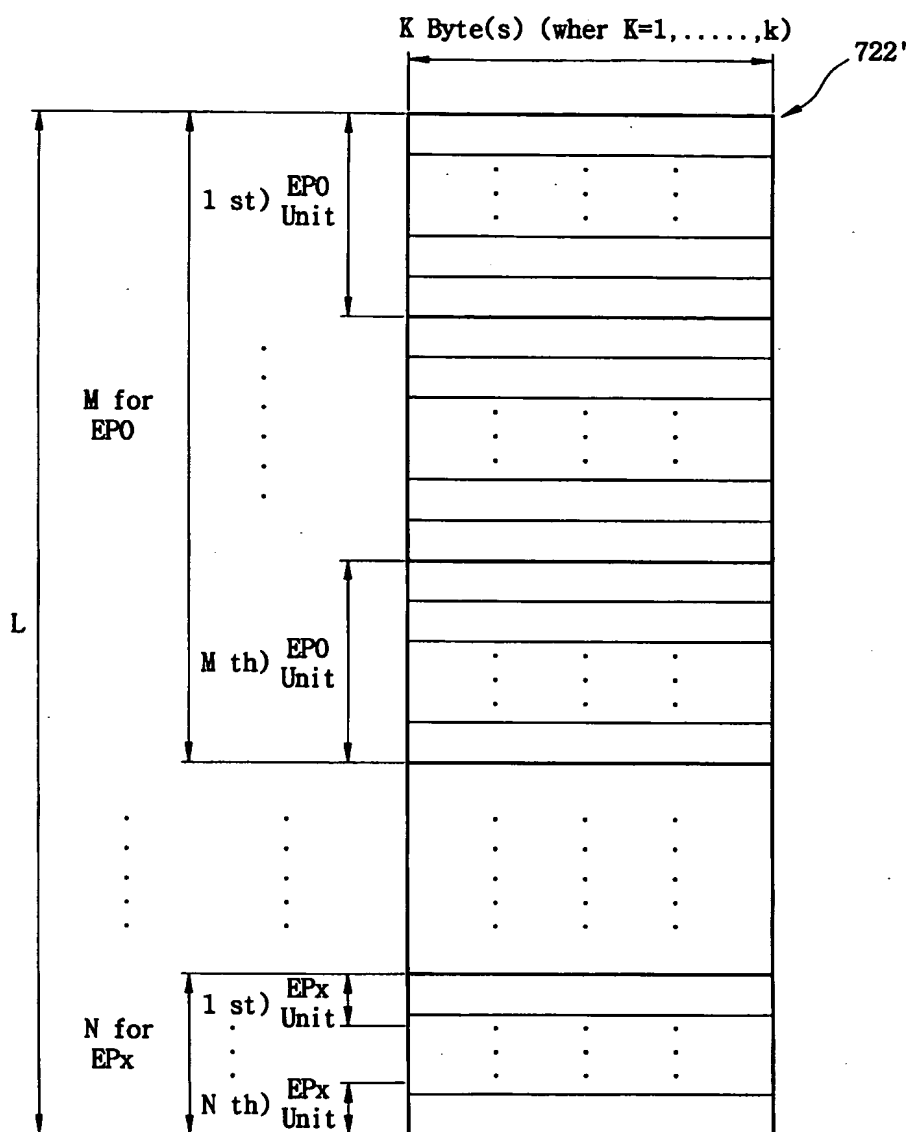
【도 8】



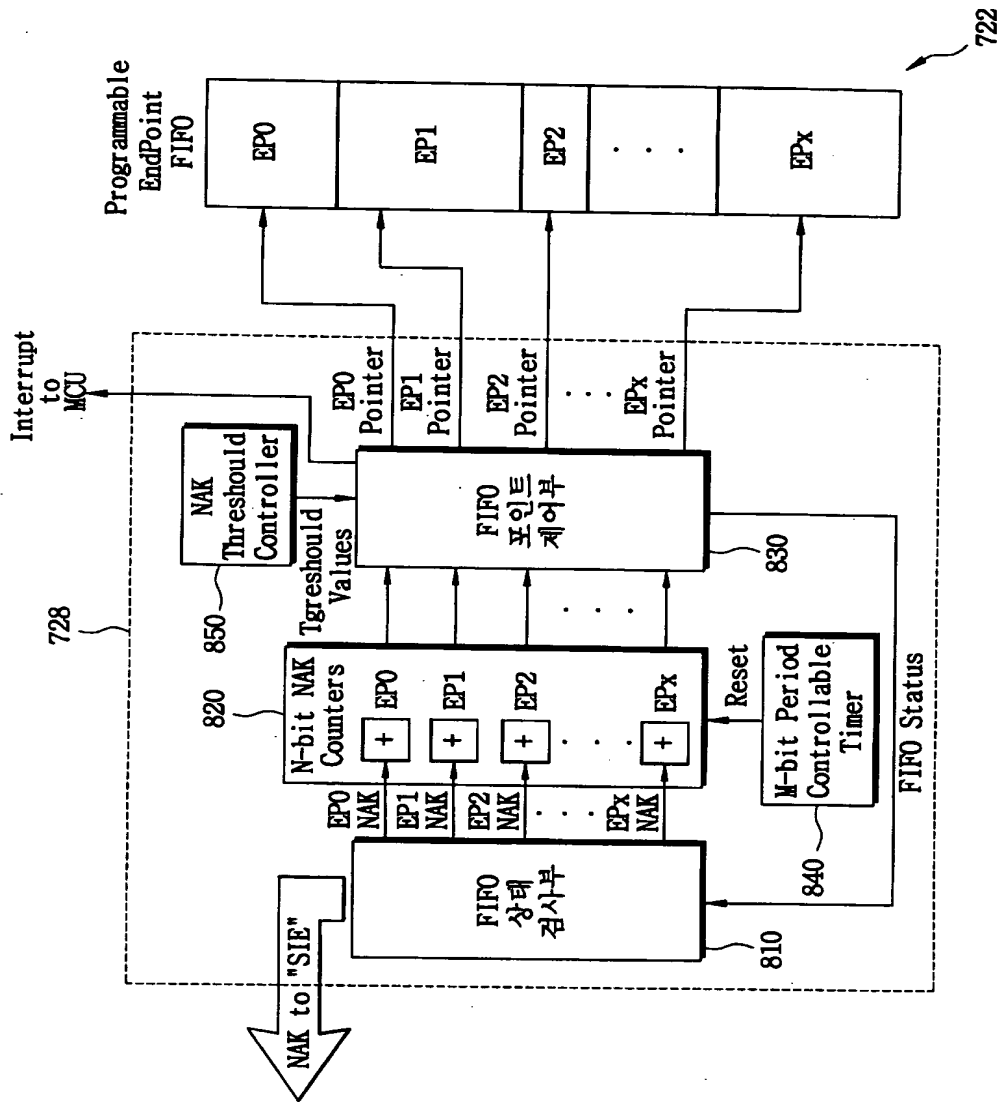
【도 9】



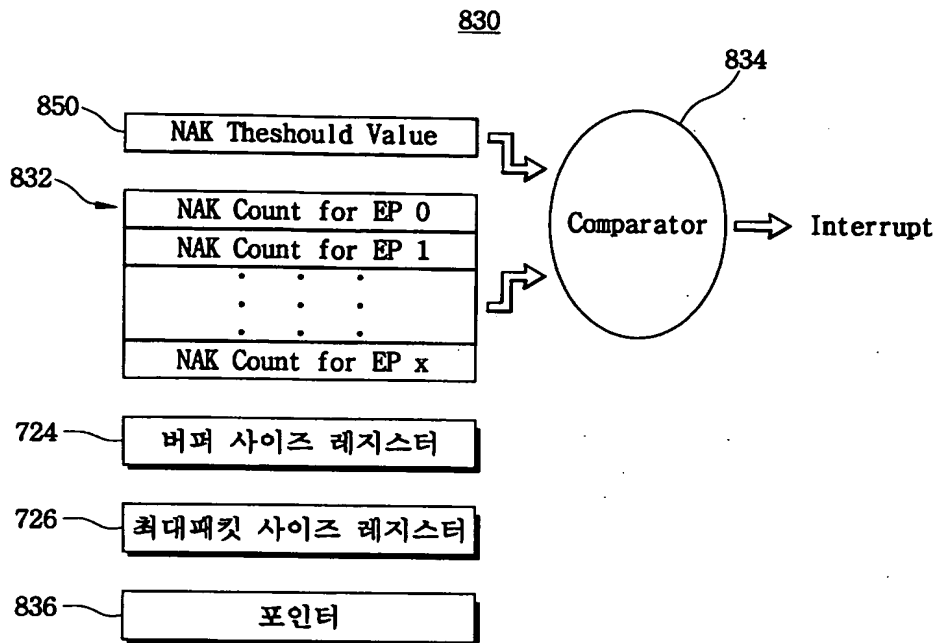
【도 10】



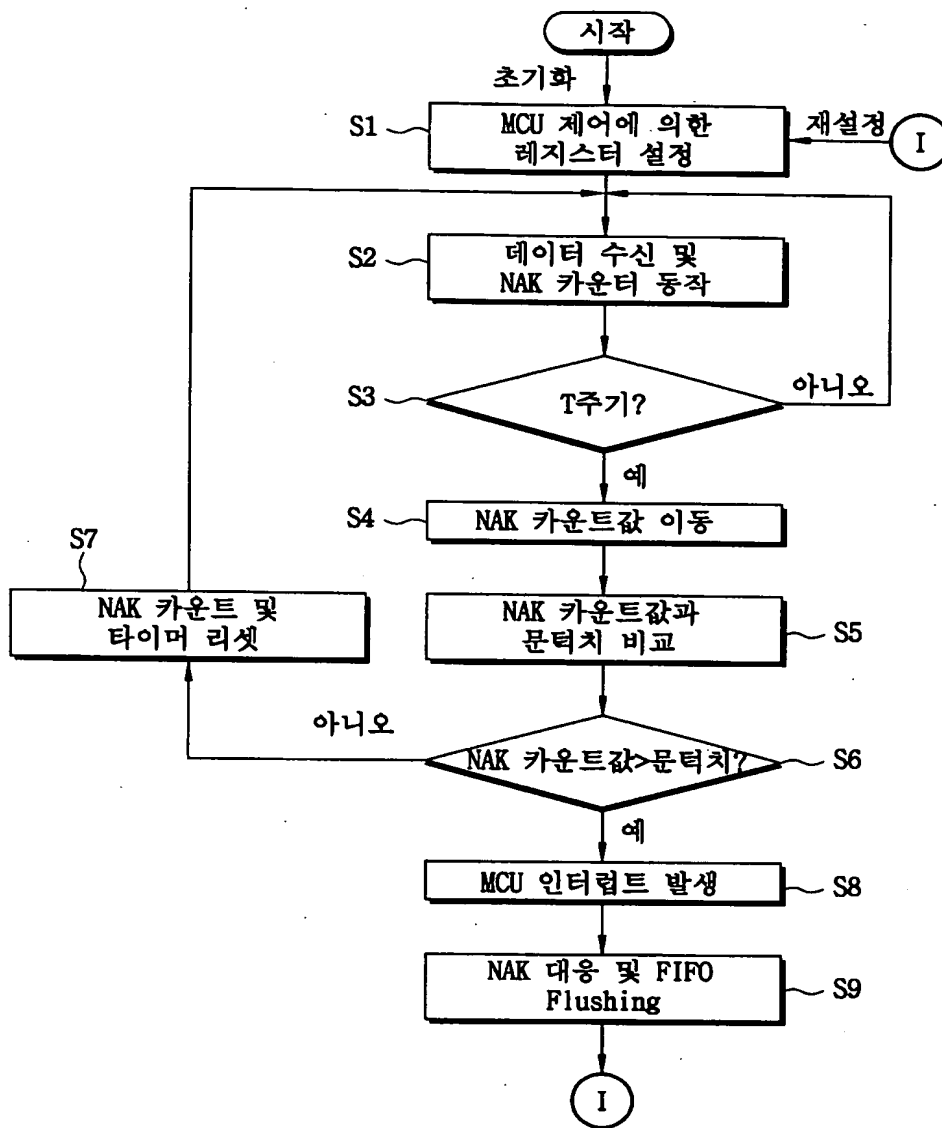
【도 11】



【도 12】



【도 13】





【도 14】

